FORTRAN Processor からみたプログラミングについての注意*

磯　田　和　男**

7090 FORTRAN で同じ hierarchy に乗除算だけがふくまれる場合の計算の順序について

7090 FORTRAN で次のような statement をかいた人がある：

K=N*100/M　　(1)

ここで K, N, M は正の整数でかつ N<M である．かいた人は

(N*100)/M　　(2)

のように演算が行なわれると思っていた．ところが実際には

(N/M)*100　　(3)

として演算されてしまうのでKの値は常に0となるわけである．

ところでマニュアルではこのようになるということは注意しないと見落しやすいが乗除算のときには (3) の例のようになる．なぜこうなるかということは FORTRAN プロセッサーについての論文 Sheridan[1] に説明されてある．これは周知のことがらとはおもうが，筆者の接触した範囲では案外しられていなかった．言語の段階だけではわからなくて，プロセッサーをしらべてはじめてわかるという一例として報告しておく．

[1] にのべられている方法で (1) の右辺を標準形になおしてみると

$$+(*(**(\oplus N))*(**(\oplus 100))/(**(\oplus M)))$$

となり，さらにこれの production を作り簡約すれば

(1, *, N)(1, *, 100)(1, /, M)

がえられる．これはただ一つの segment で構成されている．

この次に乗除のオペレーターしかふくまない Segment に対して optimization が行なわれるのであるが，そのために本文のはじめにのべたような事態がおきるのである．一般にこのような segment に対し

(i) * の数が/の数より1個だけ多いときは

//*……*/*

(ii) * の数が/の数より2個以上多いときは

**/*……*/*……*

(iii) /の数が * の数より多いときは

//……*//……/

となるように segment の成分がならべかえられる．これは AC, MQ に残された結果をうまくつかうためである．(1) については segment は

(1, *, N)(1, /, M)(1, *, 100)

となり，compile された結果は

```
    CLA   N
    LRS   35
    DVP   M
    CLM   .
    LLS   18
    MPY   2)
    ALS   17
    STO   K
2)  OCT   000144000000
```

となり，除算したときに0となってしまう．

なお文献 1) では 704 についてかかれているが，7090 でも本稿の例については同じプロセスがおこなわれているようである．

FORTRAN での Array の Subscript の処理について

次のような例があった．

A(I, J+2) のようなかたちの array でJを −1 から動かしてループを作りたい．J が Do Statement の指数と一致していないとき，650 ではできたので 7090 のときも同様にやって失敗した．

このこともマニュアルでははっきりしていないようである．そのため特にオープンのプログラマーがしばしばおかしがちのように思われる．

なぜかということはプロセッサーで考えてみると次のようになっていることからわかる．

* Remarks on Programming in view of the FORTRAN Processor, by Kazuo Isoda (Japan Atomic Energy Research Institute)

** 日本原子力研究所

Sheridan[1] によると2次元の array A (I, J) で $I=K_1\times\sum_1\pm P_1$, $J=K_2*\sum_2\pm P_2$ の形のときA (I, J) の base の番地が次のように求められる.

$$A+\varepsilon-(\pm P_1-\varepsilon_1)-\Gamma_1(\pm P_2-\varepsilon_2)$$

ただし

$$\varepsilon=\begin{cases}0 & (\text{if } \sum_1=\sum_2=\Lambda)\\ 1 & (\text{otherwise})\end{cases}$$

$$\varepsilon_k=\begin{cases}0 & (\text{if } \sum_k\neq\Lambda)\\ 1 & (\text{otherwise})\end{cases}$$

Γ_1 はAの 1 st dimension

したがって A(I, J+2) の base の番地は A の 1 st dimension を $\boldsymbol{u}_1$ として

$$A+(1-2\boldsymbol{u}_1)$$

となる. 次に I, J に対し

$$\boldsymbol{u}_1\times J-\boldsymbol{u}_1+I$$

が計算される. これは decrement part に入っているが, その部分だけがたとえば IR 1 に転送されるが符号は転送されない. そして命令の番地部は

$$A+(1-2\boldsymbol{u}_1),\ 1$$

となる. したがって I が

$$I<\boldsymbol{u}_1-\boldsymbol{u}_1\times J$$

の値をとる間, 番地修正が目的のところにいかないで, base の番地に関して対称な番地に修正されてしまうという結果になる.

参　考　文　献

1) Sheridan, P.E., The Arithmetic Translator-Compiler of the IBM FORTRAN Automatic Coding System, Comm. ACM 2 (1959), pp. 9～21.

(昭和37年11月3日受付)